LPI exam prep: Domain Name System (DNS)

Intermediate Level Administration (LPIC-2) topic 207

Skill Level: Intermediate

David Mertz (mertz@gnosis.cx)
Developer
Gnosis Software

01 Dec 2005

This is the third of seven tutorials covering intermediate network administration on Linux®. In this tutorial, David Mertz gives an introduction to DNS and discusses how to use Linux as a DNS server, chiefly using BIND 9. He shows how to set up and configure the service, how to create forward and reverse lookup zones, and how to ensure that the server is secure from attacks.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The Linux Professional Institute (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic	Learn how to configure a basic

	205): Networking configuration	TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	(This tutorial) Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server. See detailed objectives below.
Topic 208	LPI exam 202 prep (topic 208): Web services	Coming soon
Topic 210	LPI exam 202 prep (topic 210): Network client management	Coming soon
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the developerWorks tutorials for LPI exam 101. To prepare for the other exam in certification level 2, see the developerWorks tutorials for LPI exam 201. Read more about the entire set of developerWorks LPI tutorials.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Domain Name System," the third of seven tutorials covering intermediate network administration on Linux. In this tutorial, you get a solid overview of DNS fundamentals and learn how to use Linux as a DNS server. You learn about setting up and configuring a BIND server, including working with named.conf and other configuration files; you also learn about forward and reverse DNS zones, as well as the basics of DNS security, including running BIND in a

chroot jail and the DNS Security Extensions.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Domain Name System: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
2.207.1 Basic BIND 8 configuration	Weight 2	Configure BIND to function as a caching-only DNS server. This objective includes the ability to convert a BIND 4.9 named.boot file to the BIND 8.x named.conf format, and reload the DNS by using `kill` or `ndc`. This objective also includes configuring logging and options such as directoryh location for zone files.
2.207.2 Create and maintain DNS zones	Weight 3	Create a zone file for a forward or reverse zone or root-level server. This objective includes setting appropriate values for the SOA resource record, NS records, and MX records. Also included is adding hosts with A resource records and CNAME records as appropriate, adding hosts to reverse zones with PTR records, and adding the zone to the /etc/named.conf file using the `zone` statement with appropriate type, file, and masters values. You should also be able to delegate a zone to another DNS server.
2.207.3 Securing a DNS server	Weight 3	Configure BIND to run as a non-root user, and configure BIND to run in a chroot jail. This objective includes configuring DNSSEC statements such as key and trusted-keys to prevent domain spoofing. Also included is the ability to configure a split DNS configuration using the `forwarders` statement, and specifying a non-standard version number string in response to queries.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. About DNS

The Domain Name System very usefully allows users of TCP/IP applications to refer to servers by symbolic name rather than by numeric IP address. The Berkeley Internet Name Domain (BIND) software provides a server daemon called *named* that answers requests for information about the IP address assigned to a symbolic name (or a reverse lookup or other information). On the client side of the DNS system, a *resolver* is a set of libraries that applications may use to communicate with DNS servers. The BIND package comes with several client utilities to assist in configuring, querying, and debugging a BIND 9 server: `dig`, `nslookup`, `host`, and `rndc` (formerly `ndc`). In essence, these utilities call the same libraries as other client applications do, but give direct feedback on answers provided by DNS servers.

About BIND

At the time of writing, the current version of BIND is 9.3.1. The first stable version of the BIND 9 series was released in October 2000. You may find BIND 8, which is still maintained for security patches (currently at 8.4.6), on some older installations, but as a rule, upgrade to BIND 9 where possible. Truly ancient systems might have BIND 4 installed on them, but you should upgrade those as soon as possible since BIND 4 is deprecated.

All versions of BIND may be obtained from the Internet Systems Consortium (ISC; see Resources for a link). Documentation and other resources on BIND are also at that site.

Because the LPI objectives call specifically for knowledge of BIND 8 configuration, and we cover BIND 9 here, we recommend that you review the BIND 8 information on the ISC site before taking the LPI 202 exam.

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project (see Resources for a link) has a number of useful HOWTOs.

A variety of good books on Linux networking have been published, in particular O'Reilly's *TCP/IP Network Administration* is quite helpful (find whatever edition is most current when you read this; see Resources for a link).

For DNS and BIND information specifically, O'Reilly's *DNS and BIND, Fourth Edition* is a good resource (see Resources for a link); at 622 pages, it covers more detail than this tutorial can. Several other books on BIND are available from various publishers.

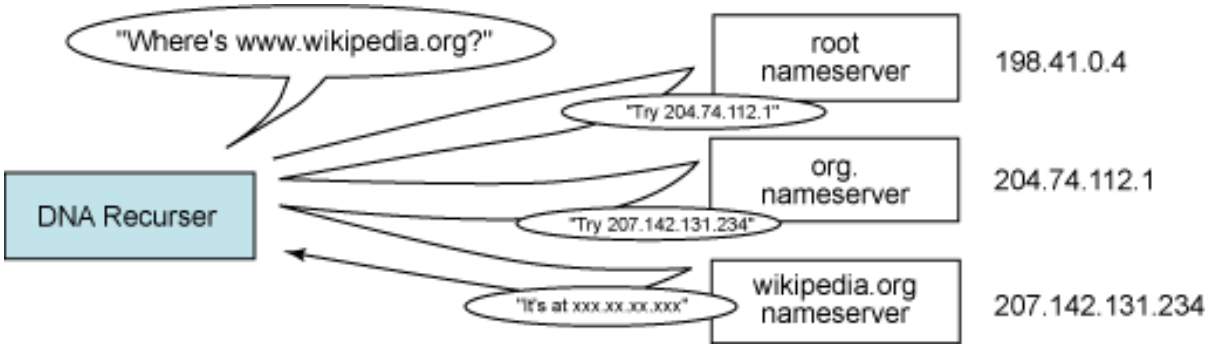
Section 3. Understanding domain name system queries

The topology of DNS

DNS is a hierarchical system of domain *zones*. Each zone provides a limited set of answers about domain name mappings, the ones within its own subdomain. A given server will query a more general server when it does not know a mapping and, if necessary, follow redirect suggestions until it finds the correct answer (or determines that no answer can be found, producing an error). When a local *named* server finds the answer to a DNS query, it caches that answer for a configurable amount of time (typically on the order of hours rather than seconds or days). By caching DNS queries, the overall network demand is lowered considerably, especially on top-level-domain (TLD) servers. The article on DNS at Wikipedia (see Resources for a link) is an excellent starting point for understanding the overall architecture. This tutorial borrows a public domain diagram from that site (see Figure 1 below).

A diagram of a hypothetical DNS query makes it easy to understand the overall lookup process. Suppose your local machine wishes to contact the symbolic domain name www.wikipedia.org. To find the corresponding IP address, your machine would first consult the local nameserver you have configured for a client machine. This local nameserver may run on the same machine as the client application; it may run on a DNS server on your local LAN; or it may be provided by your ISP. In almost all cases, it is an instance of BIND's named. This local nameserver will first check its cache, but assuming no cached information is available, will perform steps as in the following diagram:

Figure 1. Example of DNS recursion



Understand that in this diagram, the "DNS Recurser" is the actual DNS server (named), not the client application that talks to it.

DNS uses TCP and UDP on port 53 to serve requests. Almost all DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server.

How an application knows where to find a DNS server

Configuring client application access to its DNS server(s) is quite straightforward. The entire configuration is contained in the file etc/resolve.conf, whose job is principally to specify the IP addresses for one or more "local" DNS servers. You may manually configure /etc/resolve.conf with known DNS servers; however, if you use DHCP to configure a client, the DHCP handshaking process will add this information to /etc/resolve.conf automatically (you may still read it or even modify it after DHCP sets it up, but it will be reset on reboot). The library code modified by /etc/resolv.conf is called the "DNS resolver."

If more than one DNS server is configured in an /etc/resolv.conf file, secondary and tertiary DNS servers will be consulted if the primary server fails to provide an answer within the specified timeout period. A maximum of three DNS servers may be configured.

The basic entry within an /etc/resolv.conf file contains the `nameserver <IP-addr>` entries. Some other entries let you modify returned answers. For example, the directives `domain` and `search` let you expand names without dots in them (like machines on the local LAN). The `options` directive lets you change timeouts per DNS server, turn on debugging, decide when to append full domain names, and change other aspects of DNS resolver behavior. For example, one of my machines is configured with:

Listing 1. Modifying options to configure DNS servers

```
# cat /etc/resolv.conf
search gnosis.lan
nameserver 0.0.0.0
nameserver 192.168.2.1
nameserver 151.203.0.84
options timeout:3
```

The first directive lets this machine know that machines on the local LAN use the private domain *gnosis.lan*, so the simple name *bacchus* may be resolved as *bacchus.gnosis.lan*. More than one space-separated domain may be listed in the `search` directive.

Next, I list several DNS servers to try. The first is the local machine itself, which can be referred to either as *0.0.0.0* or by its official IP address, but not with a loopback address. The next `nameserver` directive lists my home-office router that connects my LAN to the Internet (and provides DHCP and DNS services). The tertiary nameserver is one provided by my ISP. I also set an option to use a three-second timeout on each nameserver rather than the default five seconds.

DNS client utilities

BIND 9 comes with four main client utilities. Three of those -- `dig`, `nslookup`, and `host` -- perform similar functions, more or less in descending order of detail. All three utilities are simply command-line utilities to exercise the DNS resolver. Essentially they do what many client applications do internally: these utilities simply provide output on the results of lookups on STDOUT. The most powerful of the three utilities, `dig`, also has the most options to limit or configure its output.

These utilities are most often used to look up an IP address from a symbolic domain name, but you may also perform reverse lookups or other record types other than default "A" records. For example, the command `host -t MX gnosis.cx` will show you mail servers associated with gnosis.cx. Some examples might help:

Listing 2. host query of google.com

```
$ host google.com
google.com has address 72.14.207.99
google.com has address 64.233.187.99
```

Listing 3. host MX query of gnosis.cx

```
$ host -t MX gnosis.cx
gnosis.cx mail is handled by 10 mail.gnosis.cx.
```

For the `nslookup` utility:

Listing 4. nslookup using default (machine-local) server

```
$ nslookup gnosis.cx
Server:         0.0.0.0
Address:        0.0.0.0#53

Non-authoritative answer:
Name:   gnosis.cx
Address: 64.41.64.172
```

Or a reverse lookup using the `dig` utility and specifying a non-default nameserver:

Listing 5. dig reverse lookup specifying a non-default nameserver

```
$ dig @192.168.2.2 -x 64.233.187.99

; <<>> DiG 9.2.4 <<>> @192.168.2.2 -x 64.233.187.99
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 3950
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;99.187.233.64.in-addr.arpa.    IN  PTR

;; AUTHORITY SECTION:
187.233.64.in-addr.arpa. 2613   IN  SOA  ns1.google.com. dns-admin.google.com.
2004041601 21600 3600 1038800 86400

;; Query time: 1 msec
;; SERVER: 192.168.2.2#53(192.168.2.2)
;; WHEN: Thu Nov 10 02:00:27 2005
;; MSG SIZE  rcvd: 104
```

The remaining BIND 9 utility to keep in mind is `rndc`. The `rndc` utility controls the operation of a name server. It supersedes the `ndc` utility that was provided in older BIND releases. If `rndc` is invoked with no command-line options or arguments, it prints a short summary of the supported commands. See the manpage for `rndc` for full information on its use.

Section 4. Basic BIND configuration and running a name server

BIND configurations

When you run the named daemon to provide a DNS server, you may choose from three modes of operation: *master*, *slave*, and *caching-only*. The named daemon itself looks in its configuration files, chiefly /etc/bind/named.conf, to determine its operating mode.

In master mode, the named server acts as the authoritative source for all information about its zone. Domain information provided by the server is loaded from a local disk file that is manually modified or updated. Each DNS zone should have exactly one master server.

In slave mode, the named server transfers its zone information from the master server for its zone. Technically, a multi-zone server can be master of one zone and slave for another, but more commonly a LAN installation has a single hierarchy between master and slave or caching-only servers. A slave server transfers complete zone information to local files from its master server, so the answers provided by a slave server are still considered authoritative.

In caching-only mode, the named server keeps no zone files. Every query relies on some other name server for an initial answer, but to minimize bandwidth usage, previous queries are cached by the caching-only server. However, any novel query must be answered by a query sent over the network. Caching-only servers are most common on local machines where client applications can often perform a lookup without any network traffic.

In the /etc/resolv.conf configuration I offered as an example earlier in Listing 1, 0.0.0.0 is a caching-only server, 192.168.2.1 is a slave server, and 151.203.0.84 is a master server. You cannot tell this for certain just based on the order or IP addresses used, but the use of the local machine pseudo-IP address 0.0.0.0 suggests that it is running a caching-only server.

Configuring named.conf

There are some standard features that pretty much every /etc/bind/named.conf file has. An initial `options` directive configures some basic information. After that, several `zone` directives provide information on how to handle various zone requests. Domains given in `zone` directives as IP addresses represent initial portions of IP address ranges, but are indicated "backwards." Symbolic names may define zones, too, allowing further specified subdomains.

named.conf files (and other BIND configuration files) follow C-like formatting conventions, in large part. Both C-style block comments (`/* comment */`) and C++-style line comments (`// comment`) are allowed, as are shell-style line comments (`# comment`). Directives are followed by semicolon-divided statements surrounded by curly brackets.

To start, here are some common options. My local /etc/bind/named.conf begins with:

Listing 6. My local named.conf starts like this

```
include "/etc/bind/named.conf.options";
```

But you may also use the `options` directive directly:

Listing 7. Configuring named.conf options

```
options {
    directory "/var/bind";
    forwarders { 192.168.2.1; 192.168.3.1};
    // forward only;
}
```

This setup lets files specified without a full path be located in a relative directory; it also tells the local named to look first in 192.168.2.1 and 192.168.3.1 for non-cached results. The `forward only` directive (commented out here) says to look only in those nameservers on the local LAN rather than ask the root servers on the Internet.

A special `zone` directive is present in nearly all named.conf files:

Listing 8. Hint zone for root servers

```
zone "." {
    type hint;
    file "/etc/bind/db.root";
};
```

The contents of `db.root` (sometimes called `named.ca` for "certifying authority") is special. It points to canonical root servers, the domain registrars themselves. Their values change rarely, but you may obtain an official latest version from ftp.rs.internic.net. This is not a file a regular administrator will modify.

Beyond the root zone hint, a named.conf file will contain some master and/or slave zones. For example, for the local loopback:

Listing 9. Loopback zone configuration

```
zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};
```

More interestingly, a named server might act as master for a domain (and provide reverse lookup):

Listing 10. External zone configuration

```
zone "example.com" {
    type master;
    file "example.com.hosts";  // file relative to /var/bind
};
// Reverse lookup for 64.41.* IP addresses (backward IP address)
zone "41.64.in-addr.arpa" {
    type master;
    file "41.64.rev";
};
```

For a slave configuration, you might instead use:

Listing 11. External zone configuration (slave)

```
zone "example.com" {
    type slave;
    file "example.com.hosts";  // file relative to /var/bind
    masters { 192.168.2.1; };
};
// Reverse lookup for 64.41.* IP addresses (backward IP address)
zone "41.64.in-addr.arpa" {
    type slave;
    file "41.64.rev";
    masters { 192.168.2.1; };
};
```

Other configuration files

The named.conf file references a number of other configuration files with the `file` directive. These names are dependent on your specific setup, but in general will contain various resource records that are defined in RFC 1033 (*Domain Administrators Operations Guide*; see Resources for a link). The standard resource records are:

SOA
Start of authority. Parameters affecting an entire zone.

NS
Nameserver. A domain's name server.

A
Address. Hostname to IP address.

PTR
Pointer. IP address to hostname.

MX
Mail exchange. Where to deliver mail for a domain.

CNAME
Canonical name. Hostname alias.

TXT
Text. Stores arbitrary values.

The format of a record is: `<name> <time-to-live> IN <type> <data>`.

The name and time-to-live are optional and default to the last values used. The literal string `IN` means Internet and is always used in practice. The resource record files may also contain directives, which begin with a dollar sign. The most common of these is probably `$TTL`, which sets a default time-to-live. For example, a somewhat trivial record file for the `127.*` localhost looks like this:

Listing 12. A trivial record file

```
# cat /etc/bind/db.127
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1     ; Serial
                         604800     ; Refresh
                          86400     ; Retry
                        2419200     ; Expire
                         604800 )   ; Negative Cache TTL
;
@       IN      NS      localhost.
1.0.0   IN      PTR     localhost.
```

Other directives are `$ORIGIN`, which sets the domain name used to complete any relative domain name; `$INCLUDE`, which reads an external file; and `$GENERATE`, which creates a series of resource records covering a range of IP addresses.

Section 5. Create and maintain DNS zones

Reverse zone files

Reverse zone files (often indicated with a .rev extension) contain mappings from zone-specific IP addresses to symbolic names. For example, you might have a file /var/bind/41.64.rev that contains:

Listing 13. Reverse zone file for 64.41.*

```
$TTL 86400
; IP address to hostname
@       IN      SOA     example.com.  mail.example.com. (
                              2001061401 ; Serial
                              21600      ; Refresh
                              1800       ; Retry
                              604800     ; Expire
                              900 )      ; Negative cach TTL

                IN      NS    ns1.example.com.
                IN      NS    ns2.example.com.
; Define names for 64.41.2.1, 64.41.2.2, etc.
1.2             IN      PTR   foo.example.com.
2.2             IN      PTR   bar.example.com.
3.2             IN      PTR   baz.example.com.
```

Forward zone files

Forward zone files (often named as *domain.hosts*) contain the crucial "A" records for mapping symbolic names to IP addresses. For example, you might have a file /var/bind/example.com.hosts that contains the following:

Listing 14. Forward zone file for example.com

```
$TTL 86400
; Hostname to IP address
@       IN      SOA     example.com.  mail.example.com. (
                              2001061401   ; Serial
                              21600        ; Refresh
                              1800         ; Retry
                              604800       ; Expire
                              900 )        ; Negative cach TTL

                IN      NS    ns1.example.com.
                IN      NS    ns2.example.com.
localhost       IN      A     127.0.0.1
foo             IN      A     64.41.2.1
```

```
www             IN      CNAME foo.example.com
bar             IN      A     64.41.2.2
bar             IN      A     64.41.2.3
```

Section 6. Securing a DNS server

Securing a DNS server

As with many services, it's a good idea to run BIND in a so-called *chroot jail*. , This limits BIND's access to other files or system resources, should a vulnerability or bug exist in BIND. Find a more detailed tutorial on running BIND with chroot at the "Chroot-BIND HOWTO" (see Resources for a link).

The general point of this procedure is that it is unwise to run BIND as the root user, or even as a common special user like "nobody." Often the user "named" is created to run BIND. The files used by this special user are placed in a local directory like /chroot/named/ and appropriate relative subdirectories.

BIND 9 provides much cleaner support for chroot restrictions than did BIND 8; a clean compile should suffice without special switches or Makefile tweaks.

DNSSEC

Beyond securing the local machine that runs BIND, it is also desirable to provide security guarantees to the DNS protocol itself. The DNS Security Extensions (DNSSEC) are a set of extensions to DNS that provide authenticity and integrity.

DNS is based on UDP rather than on TCP, and therefore does not have a mechanism for verifying a packet source. This limitation makes spoofing and interception attacks possible. In other words, DNS requesters might be fed malicious information, for example to redirect communication to an intruder's host. By adding cryptographic Transactional Signatures (TSIG) to DNS requests, DNSSEC can prevent spoofing of DNS responses. Each BIND 9 server that wishes to communicate securely must enable DNSSEC, but the enhanced protocol is otherwise backwards compatible. The first thing DNS servers must do -- those that intend to communicate securely, anyway -- is generate *key pairs*. This works in a manner very similar to SSH keys for host and server. For example:

Listing 15. Generating DNSSEC keys

```
dnssec-keygen -r /dev/urandom -a HMAC-MD5 -b 128 -n HOST \
primary-secondary.my.dom
# ls Kprimary-secondary.my.dom.*
Kprimary-secondary.my.dom.+157+46713.key
```

```
Kprimary-secondary.my.dom.+157+46713.private
```

As the filenames suggest, this generates public and private keys for the configured host, and the public key will be distributed to other servers. Get a good introduction to DNSSEC in "The Basics of DNSSEC" on the O'Reilly Network (see Resources for a link).

Resources

Learn

- Review the entire LPI exam prep tutorial series on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the LPIC Program, find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- The article on DNS at Wikipedia is a good starting point for understanding the overall architecture.
- *DNS and BIND, Fourth Edition* by Paul Albitz and Cricket Liu (O'Reilly, 2001) thoroughly covers both DNS and BIND.
- *TCP/IP Network Administration, Third Edition* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- See RFC 1033, the *Domain Administrators Operations Guide* , for the definitions of standard resource records.
- The Chroot-BIND HOWTO shows how to configure BIND to run in a chroot jail.
- The Basics of DNSSEC provides a good introduction to DNSSEC.
- View this 700 Linux User Groups around the world -- many LUGs have local and distance study groups for LPI exams.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- Find more tutorials for Linux developers in the developerWorks Linux zone.

Get products and technologies

- Download BIND from the Internet Systems Consortium.
- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with IBM trial software, available for download directly from developerWorks.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the developerWorks community by participating in developerWorks blogs.

About the author

David Mertz
David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book *Text Processing in Python* . For more on David, see his personal Web page.